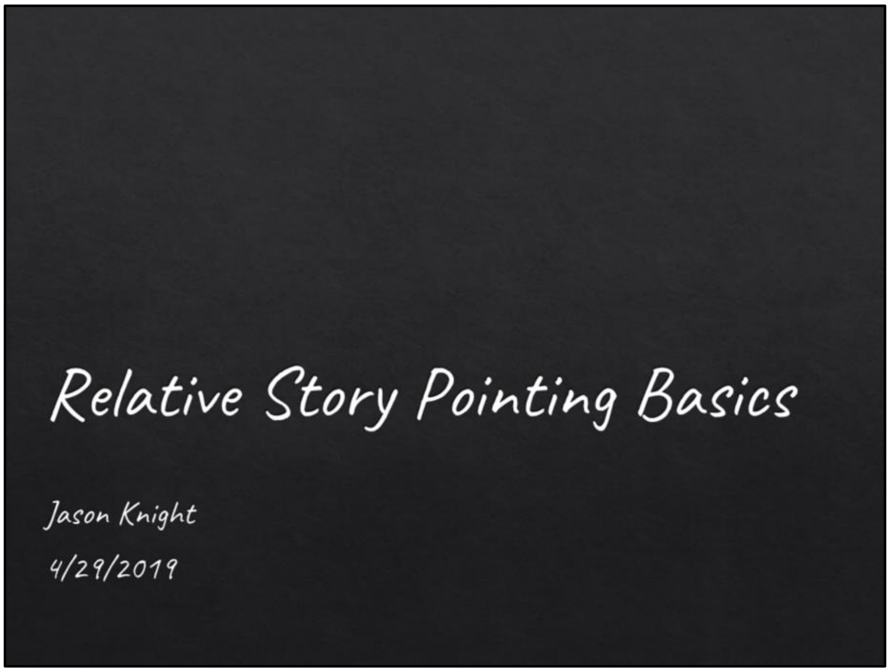
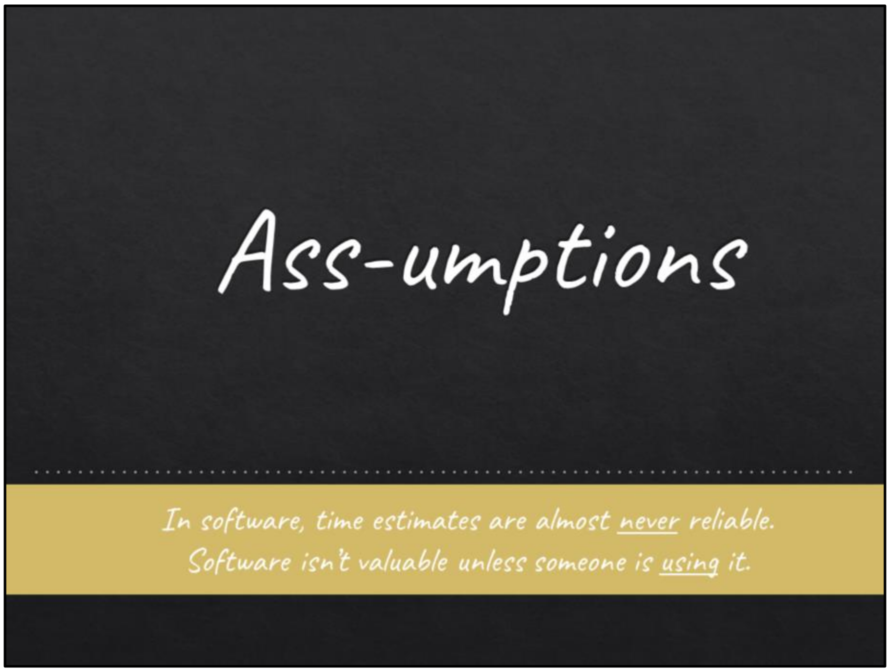


Relative Story Pointing Basics

Jason Knight

4/29/2019

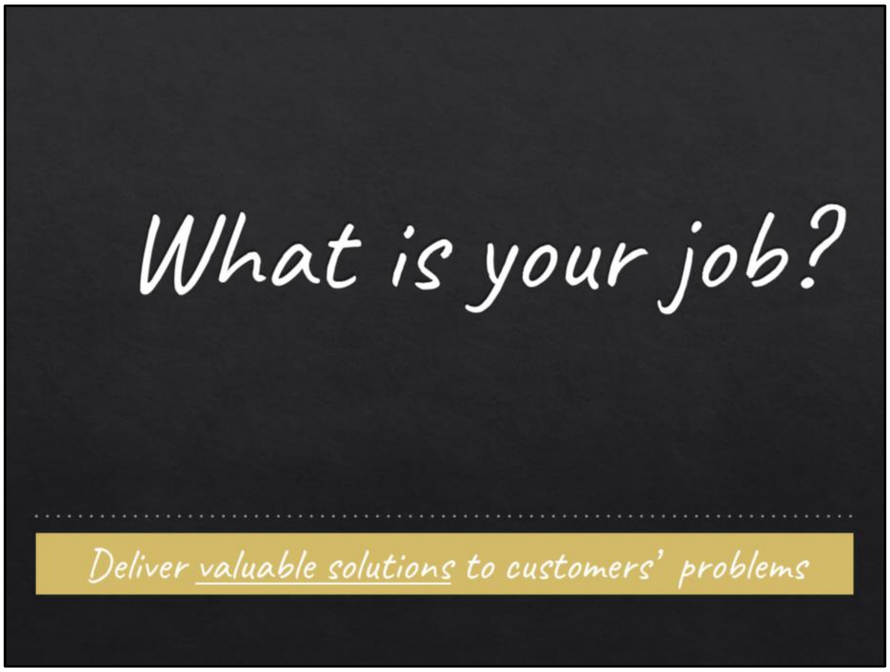
- Work estimates of time in software are almost *never* precisely predictable
- It is *more often* valuable to develop software inspecting and adapting rather than predicting and planning



Ass-umptions

In software, time estimates are almost never reliable.
Software isn't valuable unless someone is using it.

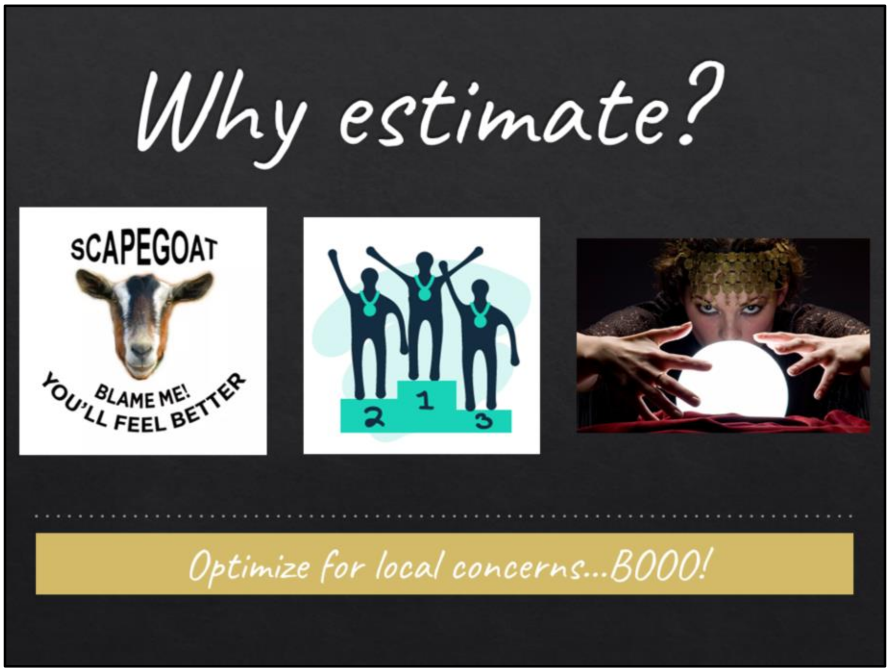
These are two assumptions I hold that factor prominently in the reasoning that follows.



What is your job?

Deliver valuable solutions to customers' problems

What is your job as a software developer? I understand that there are lots of responsibilities and activities we each choose to busy ourselves with. What is the central point of it all though?



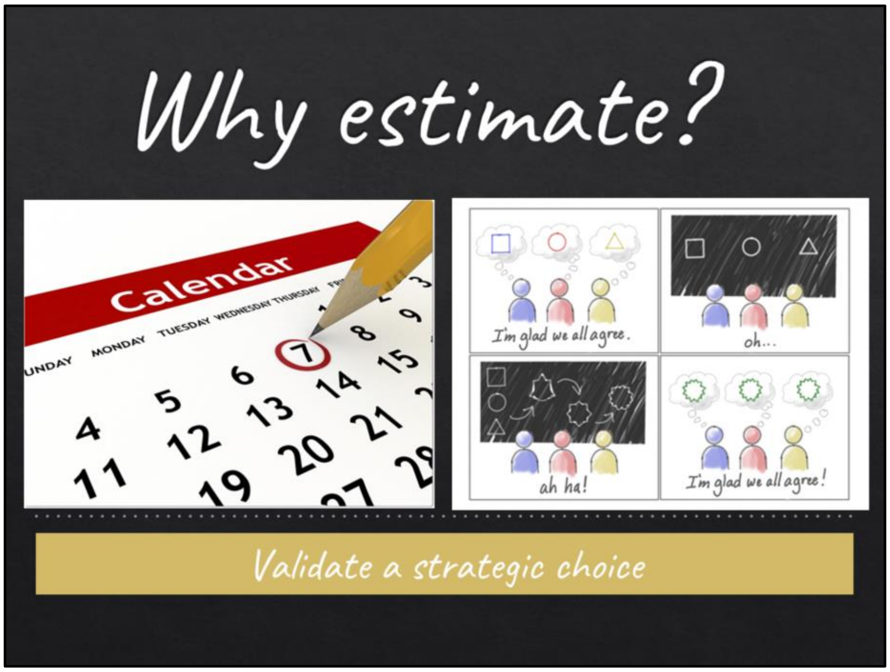
Why estimate?

SCAPEGOAT
BLAME ME!
YOU'LL FEEL BETTER
1
2
3

Optimize for local concerns...BOOO!

Bad Reasons:

- Be made a scapegoat when things go sideways
- Measure individual performance
- Predict the future for our bosses



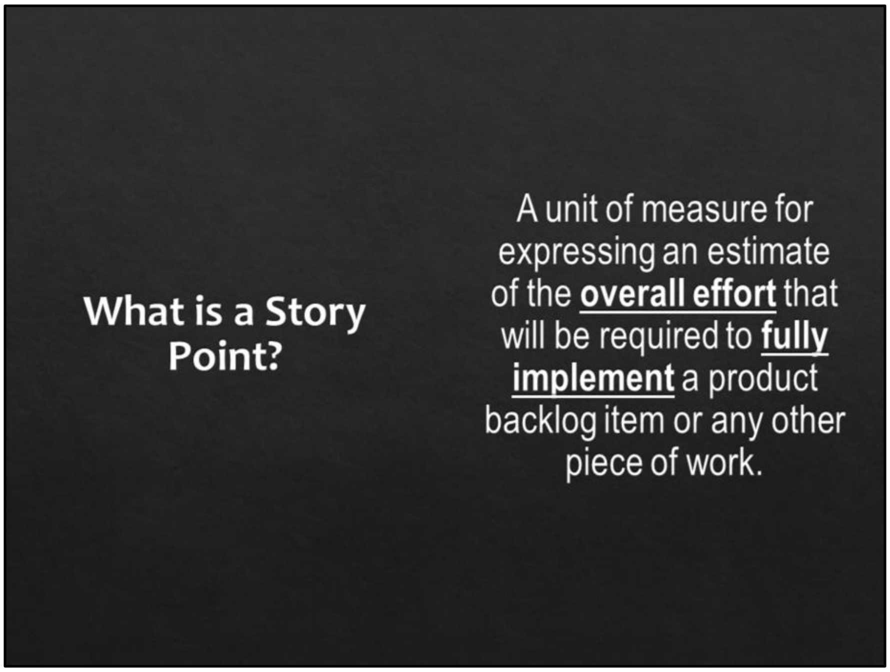
Why estimate?

Calendar
I'm glad we all agree.
oh...
ah ha!
I'm glad we all agree!

Validate a strategic choice

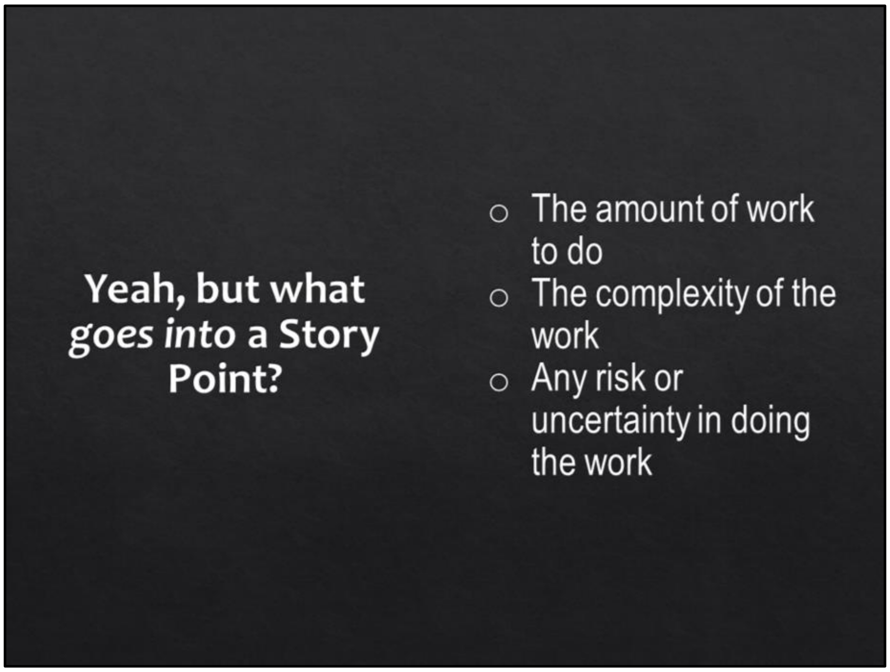
Good Reasons:

- We want to build common understanding of the work to be done soon™
- We want to forecast our ability to deliver work
- Validate strategic choice



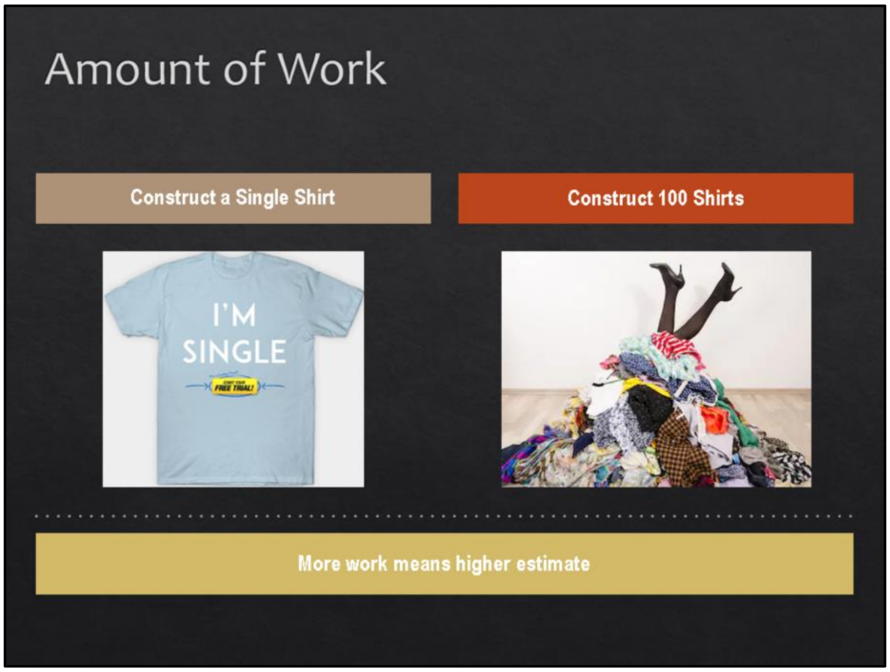
What is a Story Point?

A unit of measure for expressing an estimate of the **overall effort** that will be required to **fully implement** a product backlog item or any other piece of work.



Yeah, but what *goes into* a Story Point?

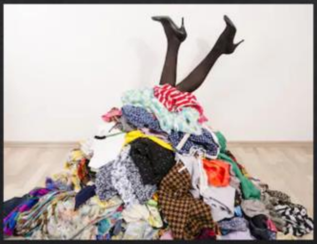
- The amount of work to do
- The complexity of the work
- Any risk or uncertainty in doing the work



Amount of Work

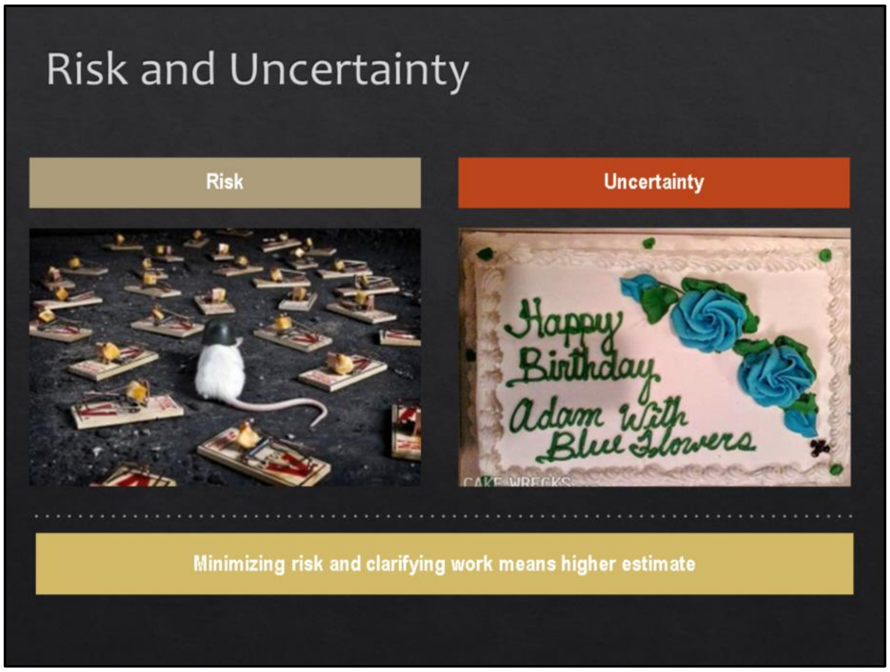
Construct a Single Shirt

Construct 100 Shirts



More work means higher estimate

As amount of work increases, estimate goes up. With a lot of work, you might find ways to do each bit faster. Not necessarily a linear progression.



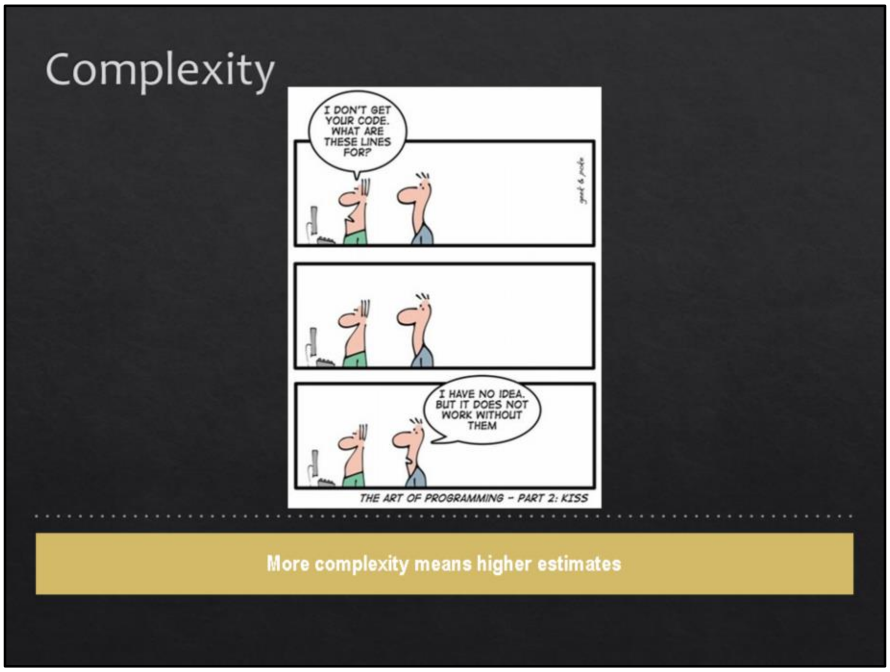
Risk and Uncertainty

Risk	Uncertainty

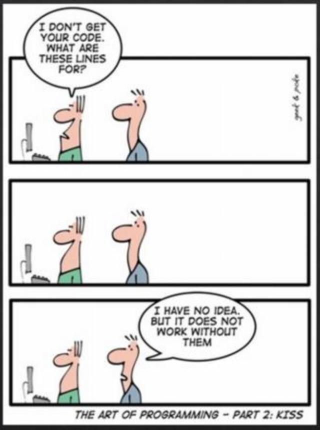
Minimizing risk and clarifying work means higher estimate

Risky work means you will proceed with more caution. Effort increases.

When work is uncertain, you will take more time to clarify. Effort increases.

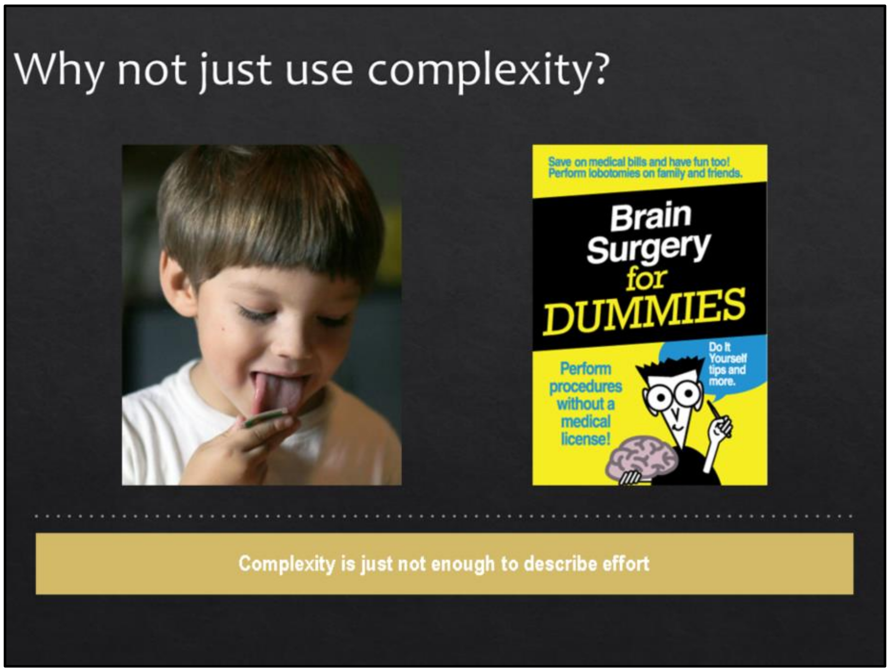


Complexity

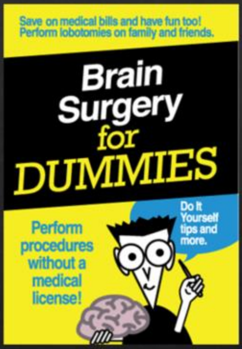

I DON'T GET YOUR CODE. WHAT ARE THESE LINES FOR?
I HAVE NO IDEA. BUT IT DOES NOT WORK WITHOUT THEM
THE ART OF PROGRAMMING – PART 2: KISS

More complexity means higher estimates

Complex work means more analysis must be performed to understand the effect of changes. Effort increases.



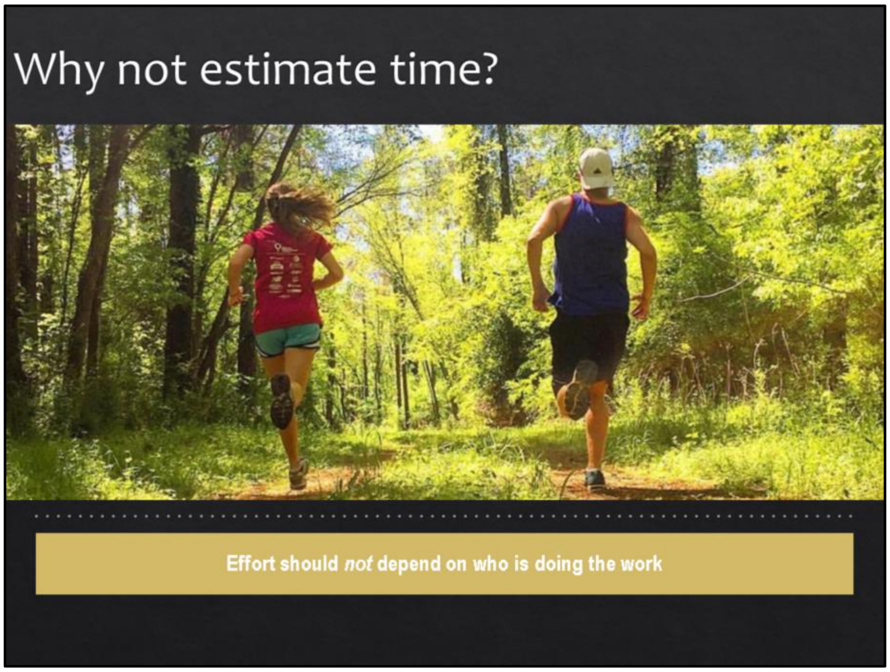
Why not just use complexity?


Save on medical bills and have fun too! Perform lobotomies on family and friends.
Brain Surgery for DUMMIES
Do It Yourself tips and more.
Perform procedures without a medical license!

Complexity is just not enough to describe effort

Which is easier. Licking 1,000 stamps or performing simply brain surgery? Presuming they take the same amount of effort, one is simple with high amount of work and the other is complex with a low amount of work.

If there is additional risk or uncertainty to either, this increases effort as well. Maybe you have a rare adhesive allergy or maybe you have low blood pressure or other pre-existing conditions. These should all increase effort as well.



Why not estimate time?

Effort should *not* depend on who is doing the work

Runner 1 runs trail in 30 minutes. Runner 2 runs it in 60 minutes. How much effort does the trail represent? A compromise of 45 serves neither. What's the answer?

It's a 5 mile trail that takes Runner 1 30 minutes AND a 5 mile trail that takes Runner 2 60 minutes. 5 miles is the abstraction that we can agree on. Without that abstraction, we can't talk about the trail in objective terms and fall prey to subjective arguments and estimates tightly-coupled to who is doing the work. Tightly-coupled work of this nature is *less* useful to the team since it likely becomes inaccurate if a different teammate decides to take on the work. You might suggest that different teammate should just re-estimate when they take the work. I suppose if that developer loves estimating, then this could be a positive for them personally. I'd argue it is a net drain on the team's time and attention given the increased information management in product backlog items.

Alternatively, if story points just map to time, wouldn't it be better to just use time?! Why waste precious hours with an utterly redundant abstraction?

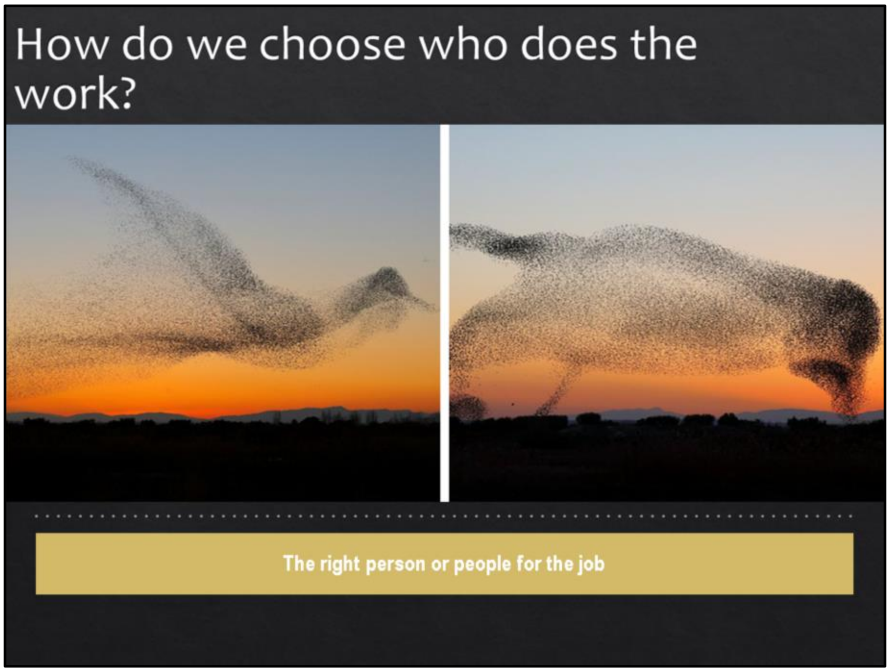
Why not forecast *individual* velocity?

Collaborative work requires collaborative forecasting

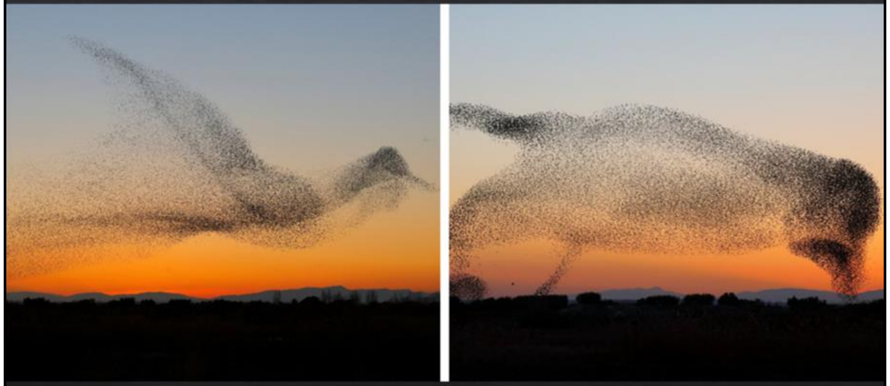
Because our work is rarely individual. If the work requires collaboration, individual velocity masks the inherent, collaborative work and leads to:

- Unnecessary complexity of forecasts
- Inaccurate forecasts
- Individual scapegoating
- Fractured focus and fractious team dynamics
- Increased time and attention paid to backlog management
- PBI's tightly coupled to individual's estimates i.e what do you do when you take on work for a teammate who had done the estimation?

Story points are instead a measure of the collectively difficulty the *focused development team* faces to complete the work.

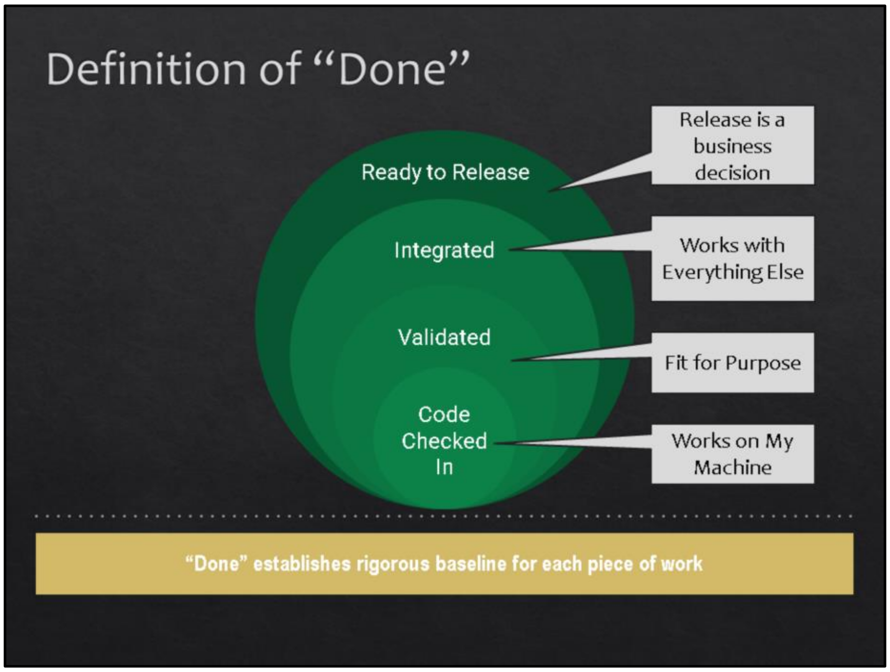


How do we choose who does the work?



The right person or people for the job

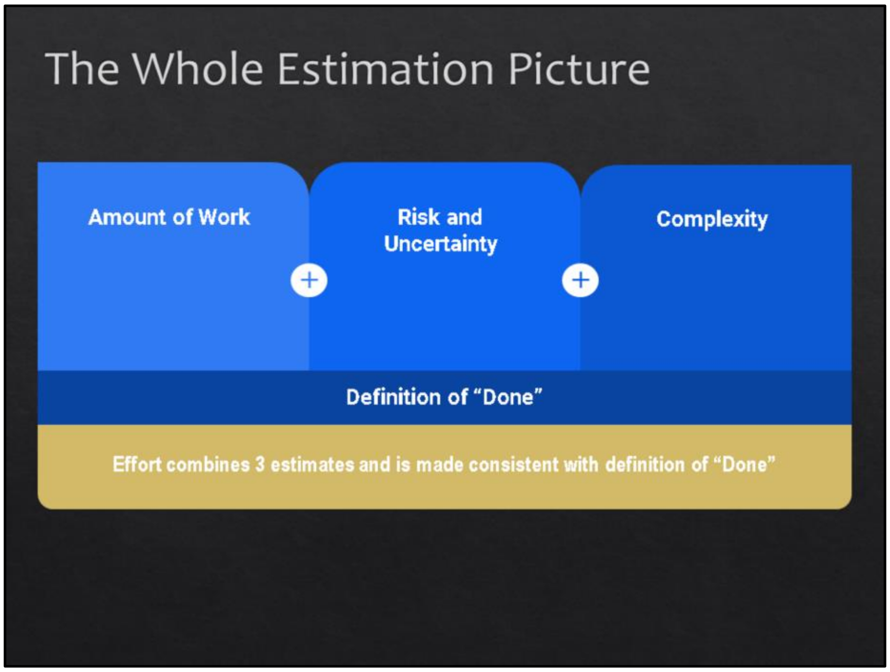
The team decides and makes their plan to get the most work done in a given time frame. This is self-organization and is the most efficient way of distributing effort. Period.



Definition of "Done"

"Done" establishes rigorous baseline for each piece of work

As a baseline, a definition of "Done" makes transparent the work needed for *all* pieces of work. In the example above, the 4 work tasks must always be done regardless of the amount, complexity and risk/uncertainty in a piece of work. If these steps are ignored or applied inconsistently, story point estimates will *estimate different things* depending on who is doing the work.

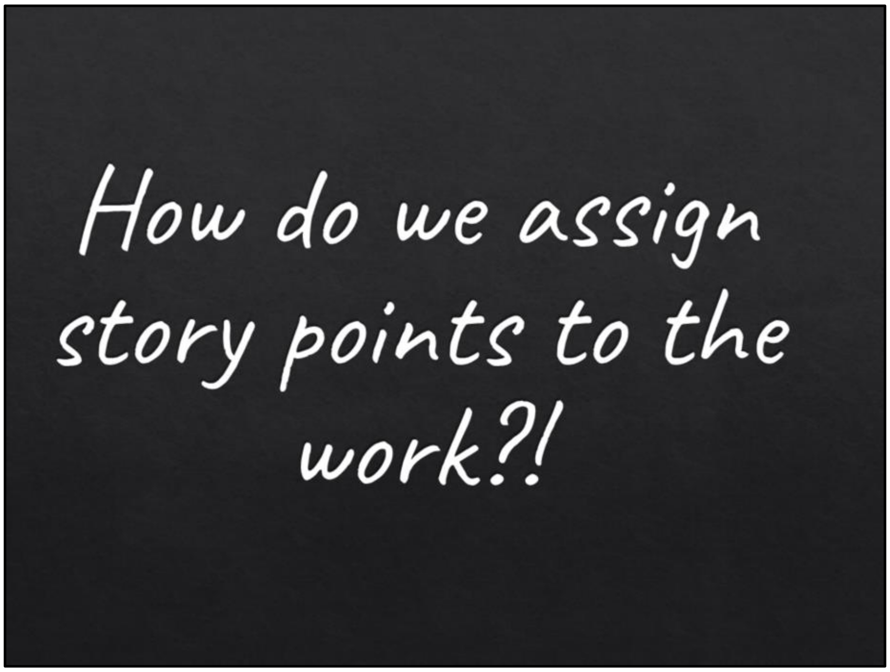


The Whole Estimation Picture

Amount of Work
+
Risk and Uncertainty
+
Complexity
Definition of "Done"
Effort combines 3 estimates and is made consistent with definition of "Done"

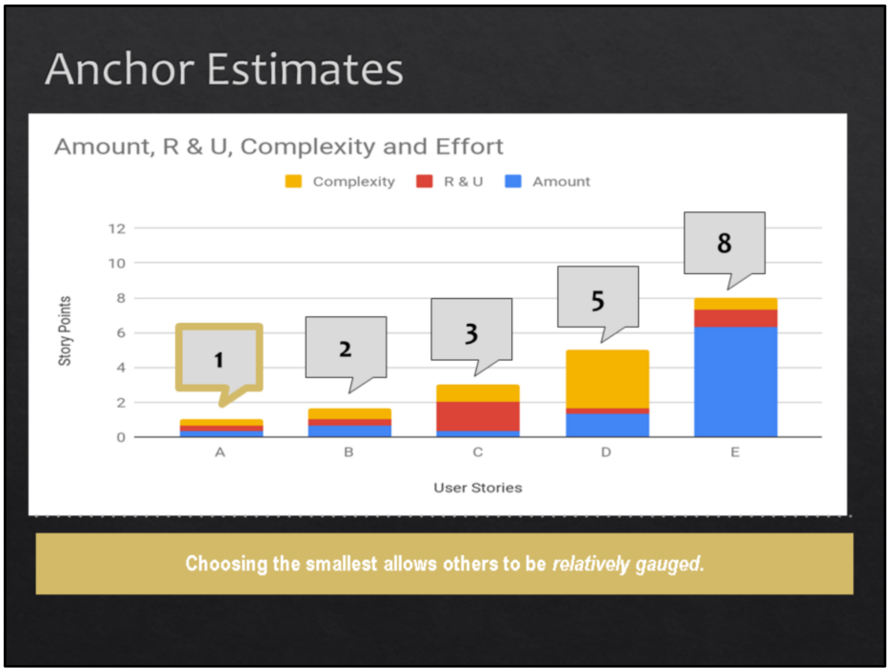
Benefits:

- All work to deliver is included
- No "hidden" work escapes the estimate; all work necessary to make something "Done" is made transparent
- Develops team's reasoning skills and estimation vocabulary
- Fosters teamliness; team thinking; collaborative approach; optimize individuals' time working.



How do we assign story points to the work?!

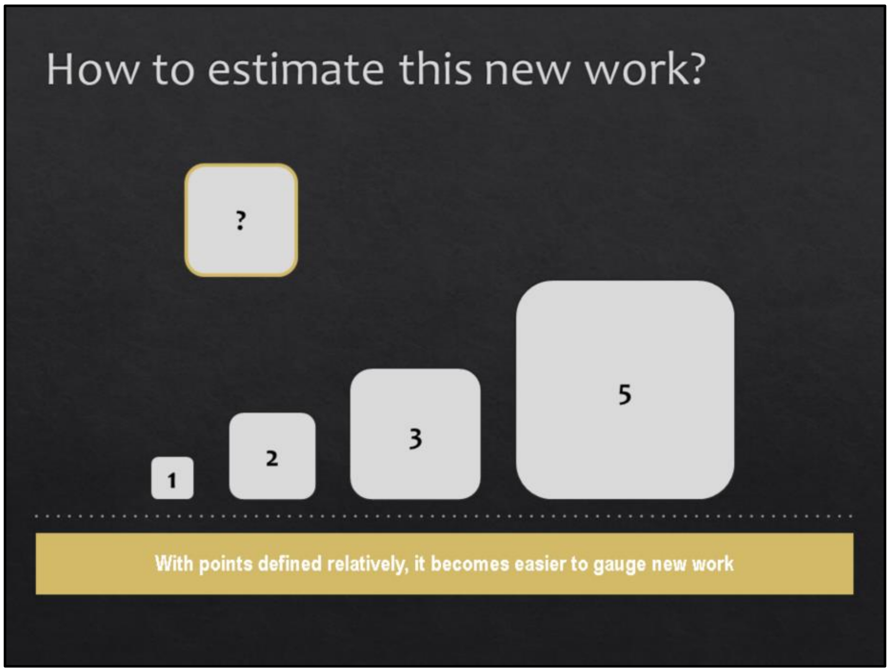
So what’s a 1 and what’s a 13? Since this is a relative scale, you must first start by creating an **anchor** against which all other work will be gauged.



Anchor Estimates

Choosing the smallest allows others to be *relatively gauged.*

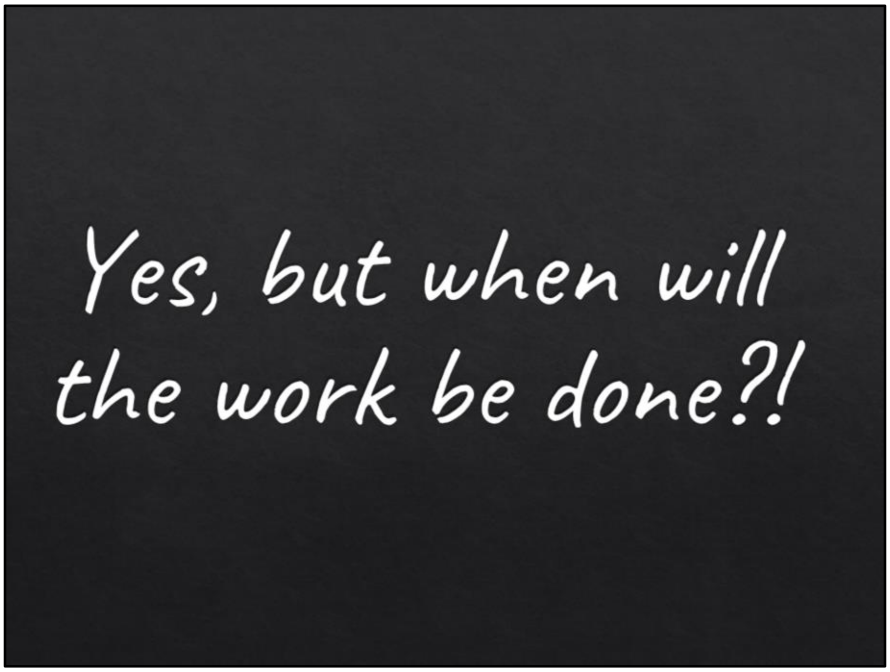
Step 1 - pick the smallest effort work and assign it a story point value of 1.
Step 2 - use a non-linear scale like a fibonacci series to make bucketing easier.
note: we're bad at estimating things as they grow in effort. A non-linear scale gives us effort distinction without forcing unnecessary precision of comparison



How to estimate this new work?

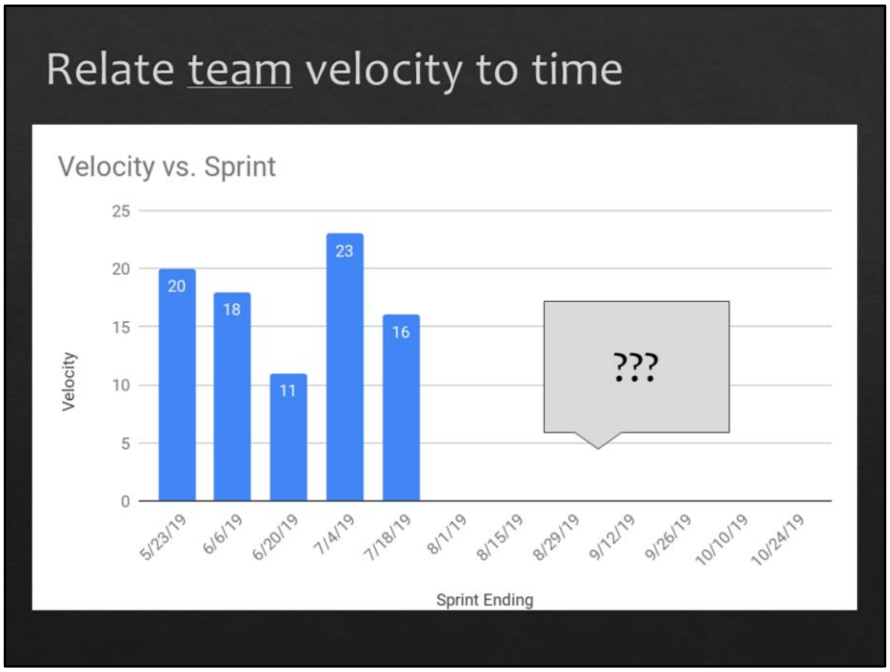
With points defined relatively, it becomes easier to gauge new work

Step 3 - place work in effort buckets, constantly relating new work to those already in buckets. This provides *accuracy* and not *precision*. Accuracy is enough.
Step 4 - Profit!! Wait, I'm getting ahead of myself...

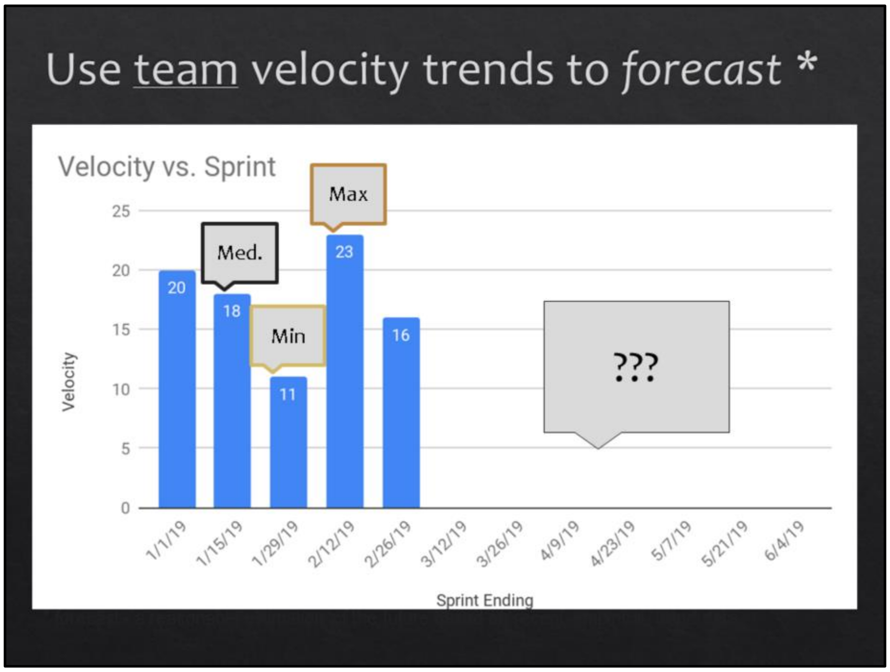


Yes, but when will the work be done?!

Story points are about time. Ok, we've said it.


Relate team velocity to time

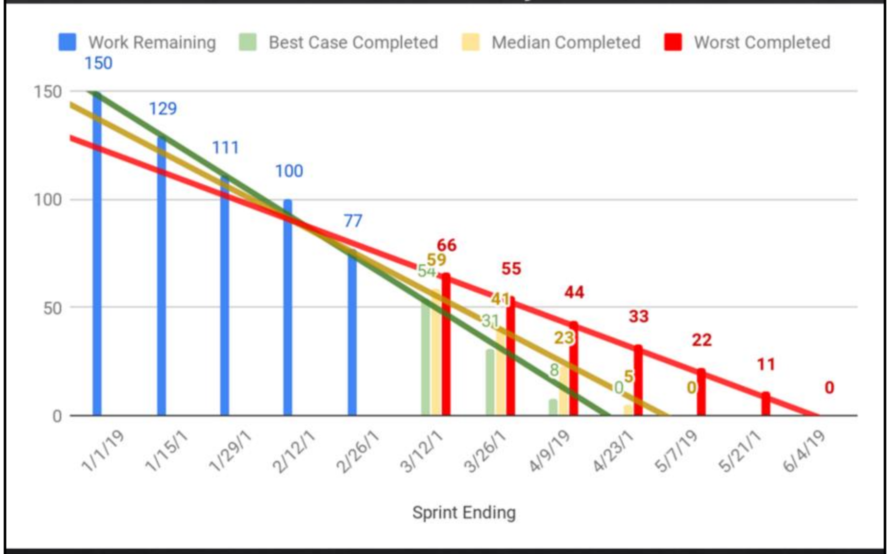
Sum the “Done” story points of work done in a Sprint timebox. Take the most recent few Sprints; historical data gets less useful over time. Yesterday’s weather is always a better metric than last year’s. What can that tell us about the future?



Using a the highest, recent velocity, a median average (not mean average) and the minimum, recent velocity forecasts can be made within the “cone of uncertainty.”

This is honest. This is empirical. This is USEFUL.

The Cone of Uncertainty

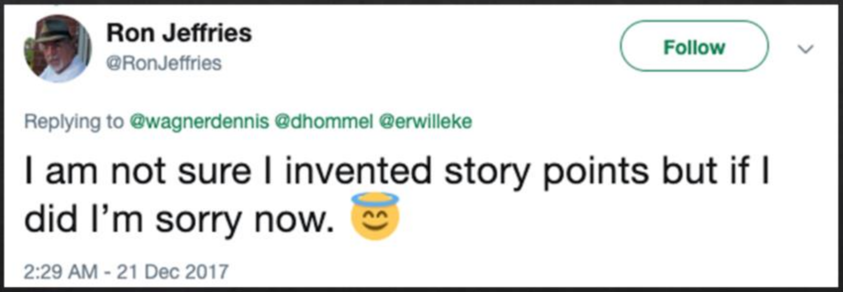


Using a the highest, recent velocity, a median average (not mean average) and the minimum, recent velocity forecasts can be made within the “cone of uncertainty.”

This is honest. This is empirical. This is USEFULL

Takeaway: honest, statistical models (even very simply ones) offer useful hints that inform our forecast

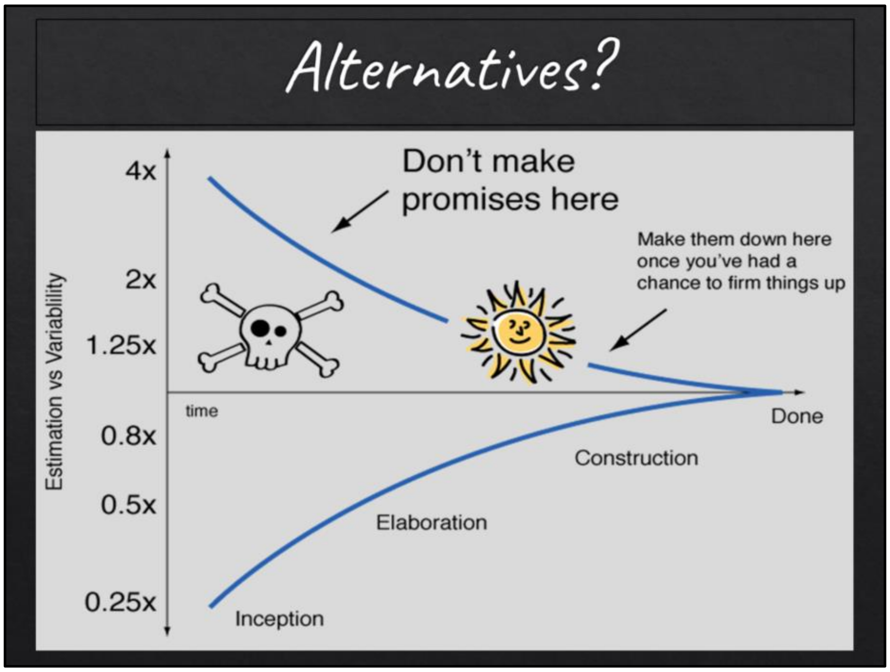
Should you use story points?


Ron Jeffries
@RonJeffries

Replying to @wagnerdennis @dhommel @erwilleke

I am not sure I invented story points but if I did I'm sorry now. 😇

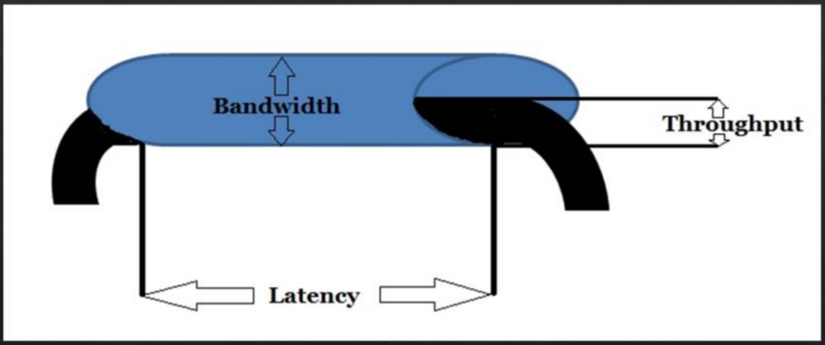
2:29 AM - 21 Dec 2017



You may be pressed to give predictions or promises for a number of important, business reasons such as:

- Executive strategies
- Sales Goals
- Account Management Concerns
- Contracting & Legal Requirements
- Project Management Plans

Alternatives?

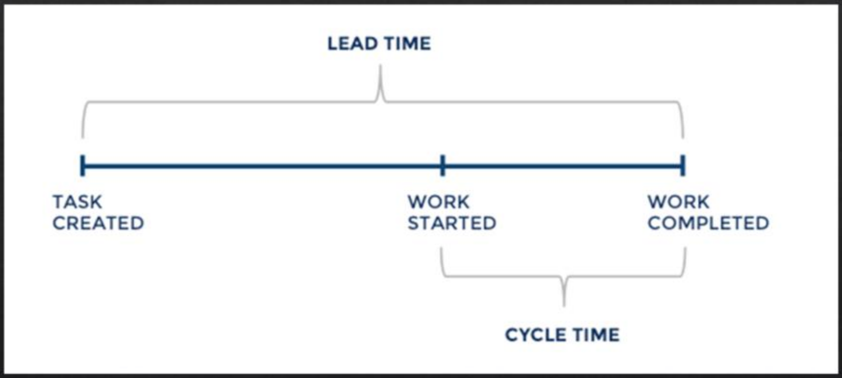


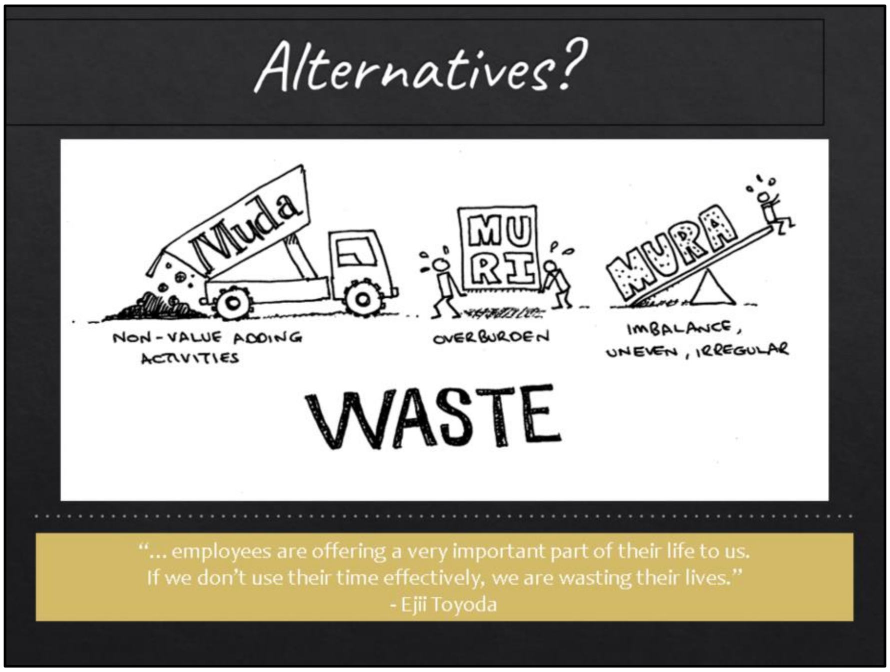
Lean Concept of Flow

Heijunka - smoothing and leveling

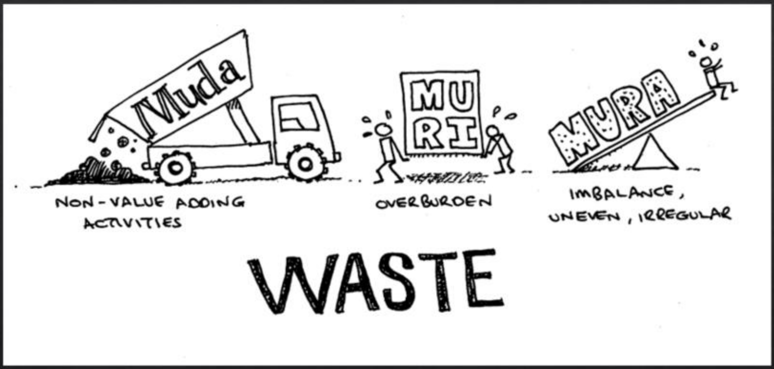
Thought experiment: if all the work was roughly the same size, would we still *need* to estimate it?

Alternatives?





Alternatives?

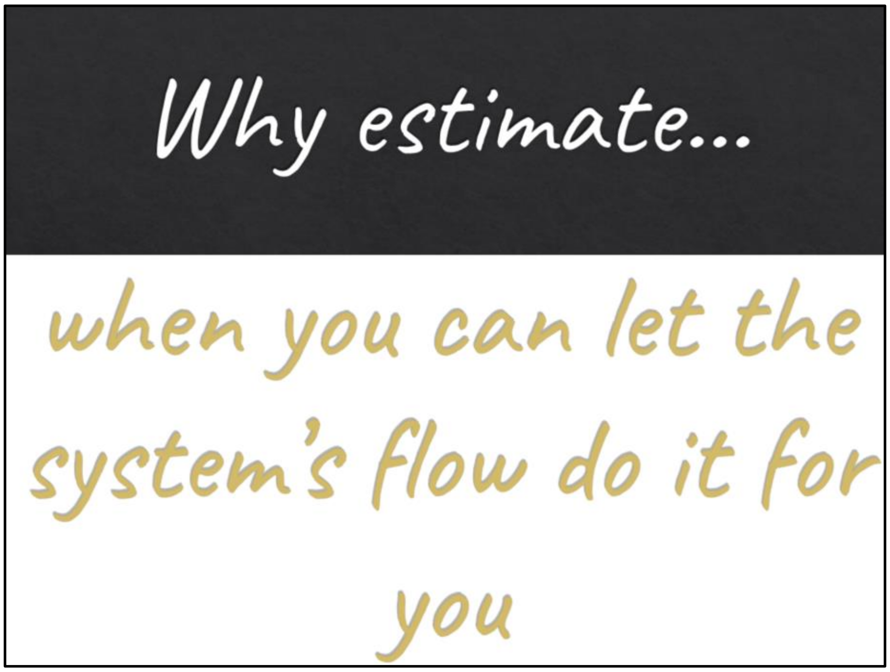


"... employees are offering a very important part of their life to us.
If we don't use their time effectively, we are wasting their lives."
- Ejii Toyoda

Work on the system, not the estimation

Identify and reduce waste in all its forms

Break down work into the smallest, most uniform bits you can



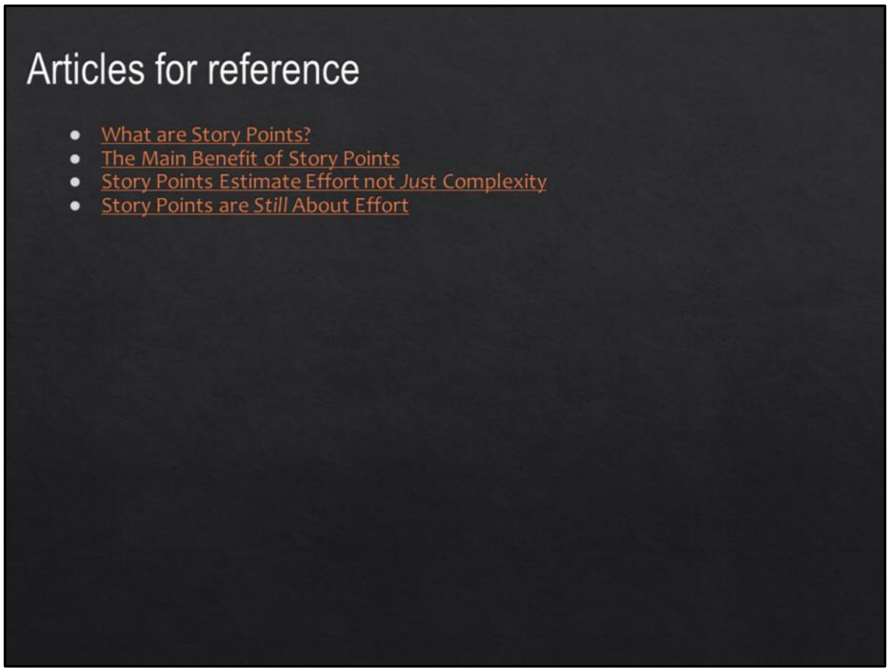
Why estimate…

when you can let the system's flow do it for you

Good Reasons:

- You'll understand the work more fully
- You'll develop a vocabulary for talking about the challenges in the work
- You'll forecast with increasing accuracy based on actual data that encompasses the *whole* amount of work
- You'll work as a team to take in, refine, complete, and deliver work with agility.

Statistical predictability of the *system's output* is a more desirable heuristic than a precise estimate.



Articles for reference

- What are Story Points?
- The Main Benefit of Story Points
- Story Points Estimate Effort not *Just* Complexity
- Story Points are *Still* About Effort